Software Safety Engineering Education

David J. Coe, Joshua S. Hogue, and Jeffrey H. Kulick
Department of Electrical and Computer Engineering, University of Alabama in Huntsville
Huntsville, Alabama, USA

Abstract – *This paper describes the Software Safety Engineering Process utilized by students enrolled in the University of Alabama in Huntsville's Software Safety Engineering course. The process consists of an industry-standard software engineering development process augmented to produce the safety-related artifacts as required for certification of a safety-critical product by a regulatory agency. The additional artifacts include Functional Hazard Assessments, Fault Tree Analyses, and Failure Modes and Effects Analyses. Students learn the impact of these new artifacts on the traditional development process as they follow our software safety engineering process to implement a representative safety-critical system, a model railroad controller.*

Keywords: software safety, DO-178B, model railroad, ARP-4761, hazard analysis, fault tree analysis

1 Introduction

Despite the ever increasing dependence on software in safety and mission critical systems, the development processes for safety critical software are frequently absent from the curricula of software engineering degree programs. A recent informal review of degree programs in the US revealed no program dedicated entirely to software safety engineering and only a smattering of courses addressing the topic. The reasons for this include the lack of appropriate text books, the tight linkage to regulatory agencies which is not normally present in academic disciplines, and the absence of good public domain examples ("go-bys") for the multitude of documents that are required for software safety engineering projects.

While safety engineering is a well established discipline with significant support in the form of textbooks, process definitions and professional societies, no such support network exists for software safety engineering. That is not to say there are not emerging standards related to constructing safe software. In fact the multiplicity of different standards is part of the problem of software safety engineering education.

The University of Alabama in Huntsville has begun development of a graduate degree program in software safety engineering within the Department of Electrical and Computer Engineering. Huntsville is well located for such a program with the abundance of organizations such as Redstone Arsenal, NASA, and aerospace companies that develop safety-critical products. This paper outlines the first course in the program, Software Safety Engineering. The course utilizes a *software safety engineering process* derived from industry-standard software engineering and safety engineering processes as well as a representative safety-critical project, a model railroad control system that must be developed to the highest level of design assurance since a train accident would be potentially fatal to humans.

2 Standard software processes

For over forty years, organizations have utilized Waterfall processes for developing large scale software systems [1]. These processes begin by identifying system-level requirements for the product and allocating those requirements to hardware or software components. Once the high-level requirements for the software components have been identified, a Software Development Plan (SDP) is constructed that identifies the artifacts that will be produced along with the development standards, supporting processes, and tools that will be used to produce those artifacts. The standard artifacts include the Software Requirements Specification (SRS), the Software Design Description (SDD), the Software Test Plan (STP), and the Software Test Description (STD) documents. The SDP also lays out a series of milestones for review of these artifacts prior to delivery to ensure that defects are identified and removed as early as possible to reduce cost and improve the quality of the product. Examples of these standard milestones include the Software Requirements Review (SWRR) and the Preliminary and Critical Design Reviews (PDR and CDR). A key component of the SDP are the rough, order-of-magnitude estimates of cost and effort required for development of the product because these estimates dictate staffing decisions and the scheduling of delivery milestones.

Standard Waterfall software development processes, however, are inadequate for the development of safety-critical systems. Systems whose failures pose a danger to life, property, or the environment often require certification by regulatory agencies who examine evidence that the system was developed with a degree of due diligence commensurate with the consequences of system failure. An integral part of this evidence is the *safety analysis* that identifies potential hazards associated with various system functions and examines the consequences of potential system failures.

The safety analysis is used to determine the degree of due diligence required during development which is called the *Design Assurance Level* or DAL. For example, in the RTCA DO-178B standard used by the Federal Aviation Administration, a system is designated as DAL A if a software fault may result in a catastrophic failure (i.e., a crash, multiple deaths) [2]. DALs B through D are reserved for systems whose failures result in less severe consequences. For instance, a system is designated as DAL B if software faults may result in a hazardous failure (i.e. significant reduction on performance or safety of system), and a system is designated DAL D if a software fault results only in a minor failure (noticeable failure with little to no safety impact).

The DAL assessment has a major impact on system cost and delivery schedule because of the additional supporting evidence that may be required by the certification agency. For each assurance level, DO-178B requires evidence that specific sets of process objectives have been completed with the number of objectives required increasing with each increase in the design assurance level. For a system assessed at DAL C, one must supply evidence that 57 objectives have been satisfied while a system assessed at DAL A must satisfy 66 objectives [2]. Furthermore, the rigor with which a given objective must be addressed varies from DAL A to DAL E. For example, there are DO178-B objectives (A7.5 – A7.7) that address structural coverage testing requirements. Modified condition-decision coverage (MC/DC) testing is mandated for DAL A systems while only statement and decision test coverage are required for DAL B and DAL C systems. Consequently, a *software safety engineering process* produces more documentation, at significant additional cost, than traditional software engineering processes.

The safety analysis, however, cannot be a one-time input to a traditional software engineering process since subsequent design and implementation choices can increase the likelihood of system failure. For example, in the systems requirements process, safety analysis may only be performed on the system functions that have been identified at that time whereas safety analysis of the design cannot occur until a design has been produced later in the development cycle. Thus, safety analysis must be an iterative activity that is integrated throughout the development process which is one fundamental difference between a traditional software engineering process and a software safety engineering process. In the following section we begin by discussing a standard safety analysis process and then describe how we have integrated safety analysis into our *software safety engineering process* as illustrated in Figure 1 below.

3 Software safety engineering process

The SAE ARP 4761 standard defines the requirements for system and software safety analysis that must be completed including an initial Functional Hazard Assessment (FHA) of the system before starting hardware/software development [3]. The FHA identifies system functions and the failure effects and conditions if the system functions were to fail. The FHA also determines the severity for the failure of each system function. For example, if the system requirement were "the system shall schedule low priority trains onto the sidings", then a failure to perform that function is a hazard that may result in a crash.

A Fault Tree Analysis (FTA) is then performed on the top-level functions identified in the FHA, breaking them down into a boolean tree of lower-level events with the lowest level named the basic events. Consider the previous example of a system hazard – failing to schedule a low priority train onto a siding. Mechanical switch failure is one example of a low-level basic event that could be the source of that hazard.

The failure effects, severity, and modes of these basic events are described within the Failure Modes and Effects Analysis (FMEA). After the completion of the FHA and FTA, the system functions are allocated to hardware or software. Depending upon the severity of those functions, independent design assurance levels (DALs) are determined for hardware and software via System Safety Analysis (SSA). For developers to accurately identify the design assurance level and subsequently conform to the objectives associated with that level, a rigorous, planned process is required since the DO-178B only provides guidance on objectives to be completed without imposing specific activities required to meet those objectives [4].

Figure 1 below shows our approach for safety-critical software development which consists of DO-178B for software development integrated with ARP-4761 for system safety. The lists of artifacts indicate when within the life cycle the artifacts were generated. Those artifacts specific to safety analysis have also been identified. End-to-end traceability is a key requirement of DO-178B.

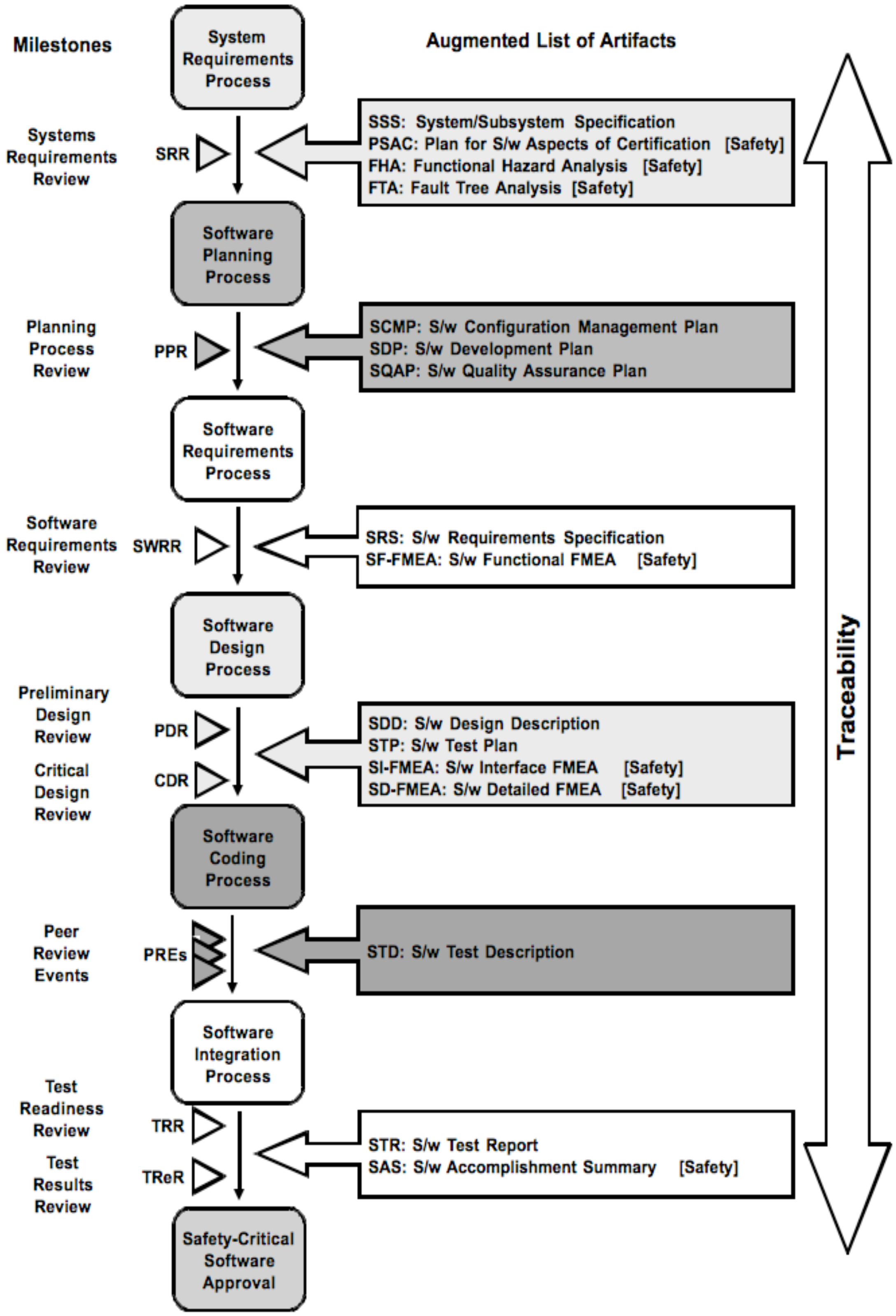

Milestones
System Requirements Process
Augmented List of Artifacts
Systems Requirements Review
SRR
SSS: System/Subsystem Specification
PSAC: Plan for S/w Aspects of Certification [Safety]
FHA: Functional Hazard Analysis [Safety]
FTA: Fault Tree Analysis [Safety]
Software Planning Process
Planning Process Review
PPR
SCMP: S/w Configuration Management Plan
SDP: S/w Development Plan
SQAP: S/w Quality Assurance Plan
Software Requirements Process
Software Requirements Review
SWRR
SRS: S/w Requirements Specification
SF-FMEA: S/w Functional FMEA [Safety]
Software Design Process
Preliminary Design Review
PDR
Critical Design Review
CDR
SDD: S/w Design Description
STP: S/w Test Plan
SI-FMEA: S/w Interface FMEA [Safety]
SD-FMEA: S/w Detailed FMEA [Safety]
Software Coding Process
Peer Review Events
PREs
STD: S/w Test Description
Software Integration Process
Test Readiness Review
TRR
Test Results Review
TReR
STR: S/w Test Report
SAS: S/w Accomplishment Summary [Safety]
Safety-Critical Software Approval
Traceability

Figure 1 – Software safety engineering process with safety artifacts identified.

Table 1 – Safety and software engineering artifacts

Safety Artifacts (ARP-4761)	Integral Process Artifacts (DO-178B)	S/w Engineering Artifacts (DO-178B)
Functional Hazard Assessment (3.2)	Verification Artifacts (11.13, 11.14, 11.17)	Planning Artifacts (11.1 – 11.5)
Fault Tree Analysis (4.1)	Configuration Management Artifacts (11.15, 11.16, 11.18)	Development Standards (11.6 – 11.8)
Failure Modes and Effects Analysis (4.2)	Quality Assurance Artifacts (11.19, 11.20)	Development Artifacts (11.9 – 11.12)

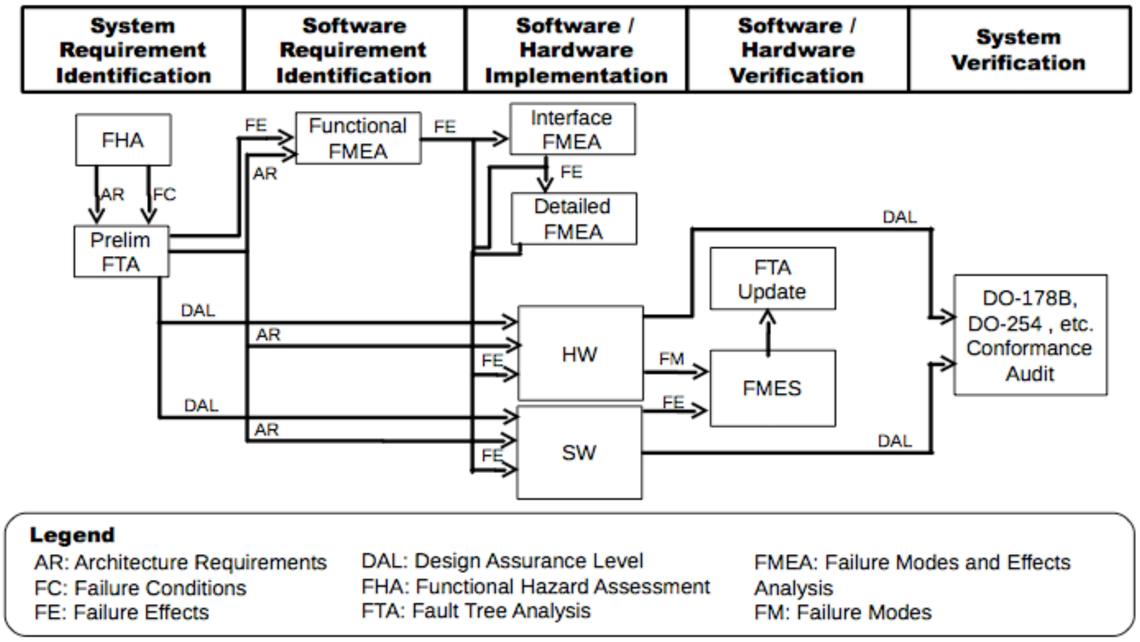

System Requirement Identification
Software Requirement Identification
Software / Hardware Implementation
Software / Hardware Verification
System Verification
FHA
AR
FC
Prelim FTA
FE
AR
Functional FMEA
FE
Interface FMEA
FE
Detailed FMEA
DAL
AR
FE
HW
DAL
AR
FE
SW
FM
FE
FMES
FTA Update
DAL
DAL
DO-178B, DO-254 , etc. Conformance Audit
Legend
AR: Architecture Requirements
FC: Failure Conditions
FE: Failure Effects
DAL: Design Assurance Level
FHA: Functional Hazard Assessment
FTA: Fault Tree Analysis
FMEA: Failure Modes and Effects Analysis
FM: Failure Modes

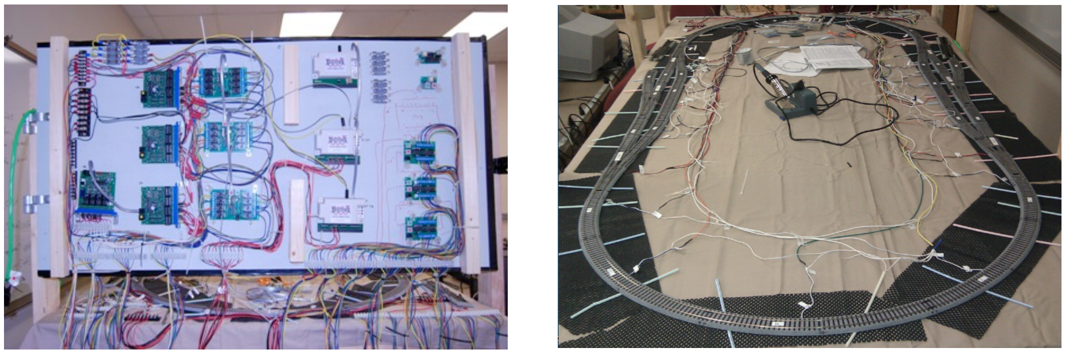
Figure 2 – Software/safety engineering process flow

The primary artifacts of interest in our Software Safety Engineering course come from several different categories, as shown in Table 1. The safety artifacts developed are based upon the SAE ARP 4761 standard, which provides guidance for FHAs, FTAs, and FMEAs in the sections shown in parentheses. DO-178B was used for development of the integral process artifacts and software engineering artifacts. The applicable DO-178B sections for the artifacts are given in parentheses.

The software engineering/safety engineering hybrid process flow integrates safety analyses in the initial determination of a DO-178B DAL through the FHA and preliminary FTA. The FMEAs are then developed simultaneously with the software requirements, design, and coding artifacts. By conducting FMEAs in the development phases, the safety analyses cover all levels of the software design and are relevant to the real system's behavior and hazards [5]. The FMEAs also provide for an indirect check on the correctness of the original software requirements, since faulty design and code are derived from faulty requirements [6]. By incorporating safety analysis throughout every phase of the software life cycle, the design team substantiates their initial DAL selection. The flow of this process is shown in Figure 2, based upon a modified hybrid of DO-178B processes and ARP-4761 diagrams for the safety assessment flow. The overview shows how the course process guarantees the software safety and software engineering activities are coalesced into a single efficient development process.

4 Model railroad system project

To fully appreciate the differences between a standard software engineering process and a software safety engineering process targeted at system certification, students must follow our software safety engineering process as they develop a safety-critical product, producing the artifacts required for certification as the project evolves. The teaching platform of choice for our Software Safety Engineering course is a model railroad project. As shown in Figure 3, the model railroad layout consists of an oval track with two sidings for passing locomotives and two dead-end spurs. A safety system is required since multiple trains traveling around the oval at different speeds and in different directions may collide.



(A) (B)

Figure 3 – Model railroad setup (A) Control logic and (B) Track layout

The primary safety system for the model railroad is a scheduling system managed by the off-the-shelf *TrainController Bronze* Scheduling Software from Freiwald Software [7]. The track layout is divided into twenty four, individually powered segments. Digitrax logic boards monitor conductive detectors within each segment to determine the segment where each train is located, and this position information is forwarded from the logic boards via USB cable to the PC-based scheduling software [8]. Similarly, commands from the scheduling software are passed via the logic boards to specific locomotives and railway switches.

Before a train may travel from one segment to another, it must reserve the next segment of track. If the next segment is available, the scheduling software sets the appropriate railway switches and allows the train to continue traveling onto the next track segment towards its destination. If the next segment is currently occupied, then the train must either stop and wait for the segment to become available or the scheduler must set a switch that diverts the train onto a siding or spur so that an on-coming train may pass.

From a teaching perspective, the model railroad project has a number of advantages. First, the baseline system is relatively low cost (approximately $5000) which is advantageous to universities. The model railroad system can also be reconfigured at modest cost to increase or decrease complexity as needed. Another important factor is that the safety hazards associated with the model railroad are easily understood and can be safely illustrated using the model railroad system. Such hazards include head-on-collisions or derailments due to incorrect railway switch settings.

5 Software safety engineering tools

A critical educational aspect of the course is the hands-on experience that students gain as they apply commercial-grade software engineering tools to generate and manage the artifacts required for safety certification. Safety certification requires the development of and strict adherence to Software Quality Assurance (SQA) and Configuration Management (CM) plans that span the entire development life cycle. These plans must address end-to-end requirements traceability, document and coding standards, structural coverage testing, and configuration and change management.

Students enrolled in the Software Safety Engineering course make use of LDRA's Testbed and TBrun for extensive static and dynamic analysis of their own code [9]. As part of the static analysis procedure, the code is audited for compliance with industry-standard, best-practice coding standards such as MISRA-C, CERT C, and JSF C++ AV standards. The LDRA tool also performs additional static analyses including code complexity analysis, reachability analysis, and data-flow analyses. Dynamic analysis of the code allows students to execute and test their code to determine the degree of structural coverage achieved -- a key feature since different design assurance levels require different levels of structural coverage such as statement coverage, branch coverage, or modified condition/decision coverage (MC/DC).

Students also utilize a combination Wind River's Simics and VxWorks tools for early on-target testing [10]. Simics is used to simulate the target platform itself. VxWorks is a real-time operating system (RTOS) used on the actual target. With student software running under VxWorks on the Simics simulator, the students gain early insight on the performance of their software on the target platform but in a simulated environment where it is fully accessible for analysis.

6 Results

As of this writing, the students have completed the system requirements process and software planning process, and they are currently working on the software

Table 2 – Functional hazard assessment results [12]

Function	Failure Effect	Severity (ARP 4761)
Locate train on track	Train location is unknown and trains may collide	Catastrophic
Power down on emergency stop	Trains cannot be stopped and may collide	Catastrophic
Accept/Verify track configuration	Trains cannot be ran on track	Minor
Accept/Verify train configuration	Trains cannot be ran on track	Minor
Control turnouts on plant	Trains cannot be safety directed and may collide	Catastrophic
Accept/Verify train schedules	Trains cannot be ran on track	Minor
Control trains according to schedule	Trains operate with limited or no control and may collide	Catastrophic

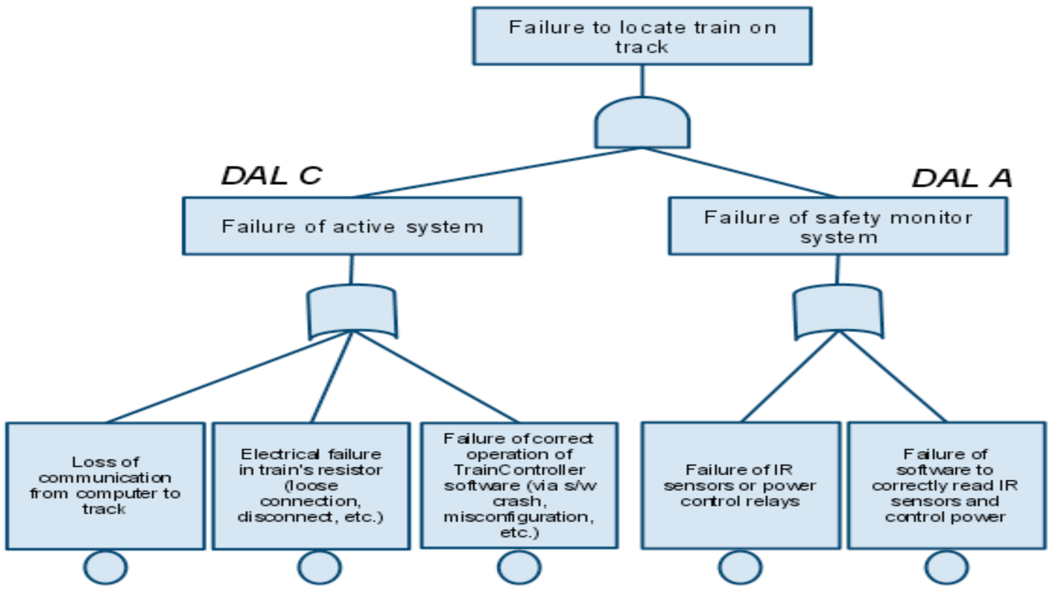

Failure to locate train on track
DAL C
DAL A
Failure of active system
Failure of safety monitor system
Loss of communication from computer to track
Electrical failure in train's resistor (loose connection, disconnect, etc.)
Failure of correct operation of TrainController software (via s/w crash, misconfiguration, etc.)
Failure of IR sensors or power control relays
Failure of software to correctly read IR sensors and control power

Figure 4 – Fault tree analysis for "Locate Train on Track" function [12]

requirements process artifacts. Since they have not completed the development of the full model railroad system, only initial results are available. However, even the initial results provide an insight into the advantages of our software safety engineering process.

The students set out to design the complete train scheduling system around the off-the-shelf Scheduling Software. The students' initial system requirements specification and FHA identified seven system-level functions of the Scheduling Software and also identified the failure effects of those functions and the severity classification of the failure effects. The system functions, failure effects, and severities are shown in Table 2.

Using the FHA analysis, the students ascertained that the Scheduling Software must meet DO-178B level A, since the worst case failure of the Scheduling Software would lead to the catastrophic event of two trains colliding. After evaluating the requirements for DO-178B level A, the students decided that developing the model railroad system to satisfy DAL A would be extremely difficult due to the complexity of the Scheduling Software and the lack of source code for that software. As a result of their safety analysis, students devised an *amelioration plan* in which they would develop a *Safety Monitor* system to handle the safety-critical aspects of the system independently of the Scheduling Software.

The *Safety Monitor* is responsible for continuously monitoring train position and cutting power to the tracks to prevent catastrophic events such as collisions from occurring. The Safety Monitor system consists of CTI Electronics optical sensors for track segment entry/exit detection [11], power control modules, and a single monitoring software procedure. Unlike the Scheduling Software which may be thousands of lines of code, the Safety Monitor only has to observe adjacent track segments and shut off power if a collision is eminent. As a result of this simplicity, students determined that it would be easier to develop the Safety Monitor to satisfy

DO-178B DAL A standard than to demonstrate that the more complex, off-the-shelf Scheduling Software satisfies DAL A.

Students demonstrated the efficacy of this Safety Monitor approach through the FTA of the system functions. As illustrated in Figure 4, a catastrophic severity function (level A) is comprised of the "active system" (Scheduling Software) providing the function capability at DO-178B level C and a Safety Monitor providing the safety-critical protection of the system at DO-178B level A. A failure will only occur through the failure of both the Scheduling Software and the Safety Monitor. By introducing the Safety Monitor system, the Scheduling Software's required assurance level was dropped from A to C, drastically reducing the number of DO-178B objectives and activities that must be completed with independence.

7 Conclusions

As software continues to become an integral component of more and more products whose failures may prove catastrophic, the demand for instruction in software safety engineering processes will increase. One of the challenges in offering our software safety engineering course has been the general lack of understanding among software developers regarding safety analysis and how a software safety engineering process differs from standard software development processes. It has become clear that while there are numerous software engineering courses and texts that discuss standard software processes and specific development activities such as requirements elicitation and testing, there are few texts that discuss the integration of safety analysis into the software development process and how safety certification requirements impact the artifacts that must be delivered. Our use of the model railroad project in our Software Safety Engineering course has proven to be a very effective means of conveying the nuances of software safety engineering to our students when there are few other resources readily available.

8 Acknowledgements

The authors would like to acknowledge the following contributions. George Petznick for assistance with wiring. James Norris of Redstone Arsenal Model Railway Club for model railway discussions. LDRA and WindRiver for software donations. Todd White from FAA Consultants for document templates. Steve Hosner for guest lecture on Fault Tree Analysis.

The authors are especially grateful to Elise Haley, Joshua Hogue, Michael Gallaher, and Hunter Stinson for granting us permission to include a portion of their FHA and FTA in this manuscript.

9 References

[1] W. R. Royce, "Managing the Development of Large Software Systems," *Proc., IEEE WESCON,* 26, 1970, pp. 1-9.

[2] RTCA DO-178B, Internet: http://www.rtca.org/

[3] SAE ARP-4761, Internet: http://www.sae.org/

[4] C. Bertrand, C. Fuhrman. "Towards defining software development processes in DO-178B with openup." *Canadian Conference on Computer and Electrical Engineering*, 2008.

[5] K. Allenby, T. Kelly. "Deriving Safety Requirements Using Scenarios." *Fifth IEEE International Symposium on Requirements Engineering*, 2001.

[6] A. Arkusinski. "A Method to Increase the Design Assurance Level of Software by Means of FMEA." *The 24th Digital Avionics Systems Conference*, 2005.

[7] "Friewald Software." Internet: http://www.friewald.com.

[8] "Digitrax." Internet: http://www.digitrax.com

[9] "LDRA Software Technology Homepage." Internet: http://www.ldra.com.

[10] "Wind River Simics." Internet: http://www.windriver.com/products/simics/.

[11] "CTI Electronics." Internet: http://www.cti-electronics.com.

[12] E. Haley, J. Hogue, M. Gallaher, and H. Stinson, Safe Train Project Documentation, unpublished.